暗号化データベースモデルにおける関係情報推定を防ぐ索引手法

AUTHOR(S):

川本, 淳平; 吉川, 正俊

SCIS 2010 The 2010 Symposium on Cryptography and Information Security
Takamatsu, Japan, Jan. 19-22, 2010
The Institute of Electronics, Information and Communication Engineers

暗号化データベースモデルにおける関係情報推定を防ぐ索引手法
A Link Disclosure Free Index on Encrypted Database Model

川本 淳平* 吉川 正俊†
Junpei Kawamoto Masatoshi Yoshikawa

あらまし 本稿では，暗号化データベースモデルにおける関係情報推定を防ぐ新しい索引手法を提案する．従来の索引手法では，共通の条件で問い合わせた利用者集合を求めることが可能なため，問合せから利用者間の関係が計算されうるという問題があった．本稿で提案する新しい索引手法では，問合せをベクトルによって表すことで，一つの問合せ条件に対して異なる複数の問合せ表現を対応させている．その結果，問合せから共通の条件で問い合わせた利用者集合を計算することが現実的には不可能となり，利用者間の関係計算を防ぐことができる．

キーワード 暗号化データベース，Link disclosure, 索引

1 はじめに

クラウドコンピューティングというフレームワークは，従来独自に運用するにはコストが高かった様々な機能，例えばデータベースや Web サーバ，仮想計算機をサービスという形式で提供している．開発者は，これらのサービスを組み合わせて使用することで，安価に新しいアプリケーションを開発することができる．しかし，クラウドサービスを利用するアプリケーションでは，利用者のデータがどのように扱われているのか分かりにくく，通常のアプリケーションに比べてプライバシ漏洩のリスクが高くなるといった問題がある．そのため，アプリケーション開発者は，プライバシに配慮したクラウドサービスを使用するように心がけねばならない．逆に，クラウドサービスの提供者にとっては，セキュリティの高さは付加価値となる．

特に，クラウドサービスの中でも中心的な役割を果たし，多くの情報を保存するデータベースサービスではセキュリティ問題は特に重要である．データベースサービスが情報漏洩の原因とならないことを示すために，攻撃者だけでなくサービスプロバイダさえ保存されている情報にアクセスできないことを保証する必要がある．この要求を満足するものとして，暗号化データベースモデル (EDB モデル)[2] が提案されている．EDB モデルでは，スキーマ $R(A_1, A_2, \cdots, A_n)$ にしたがうタプル t は，クライアント上で予め暗号化されサーバ上ではスキーマ $R_s(etuple, I_1, I_2, \cdots, I_n)$ にしたがうタプル t_s として保存される．ここで，$etuple$ は元々のタプル t を暗号化したものであり，A_i と I_i はそれぞれ元のスキーマにおける属性とその属性に対応する索引情報である．索引とは，暗号化されたタプル t_s の問合せ処理に用いられる情報であり，EDB モデルにおけるクライアントは問合せ条件としてこの索引値を用いてサーバへ問い合わせる．索引値から本来の値が推測できないことが必要であり，簡単に計算できる一方範囲問合せが行えず限定的な用途でしか使用できないハッシュ索引 [5] や，一致問合せや範囲問合せなど汎用的な用途に使用できるバケット索引 [2, 3] などが提案されている．このモデルを使用することで，サーバへ保存されるデータは暗号化され，攻撃者だけでなくサービスプロバイダさえも閲覧不可能となる．

EDB モデルを使用することで，タプルの中身については閲覧されないことが保証できる．しかし，攻撃者やサービスプロバイダが閲覧できる情報はそれだけではない．特に，Web 検索における検索語の場合 [1] と同様に，サービスを利用する際に必要となる問合せにも重要な情報が含まれている．例えば，ある数名の利用者だけが特定の問合せを使用したとすると，この利用者の間には何らかの関連があると推定できる．これはつまり，利用者からの問合せを調べることで Link disclosure [4] が起こり得ることを表している．Link disclosure とは，利用者間の関係が漏洩し，さらには利用者間のグループ構造が

* 日本学術振興会特別研究員, 京都大学大学院情報学研究科 〒 606-8501 京都市左京区吉田本町. Department of Social Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan. j.kawamoto@db.soc.i.kyoto-u.ac.jp
† 京都大学大学院情報学研究科 〒 606-8501 京都市左京区吉田本町. Department of Social Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan. yoshikawa@i.kyoto-u.ac.jp

漏洩してしまう問題である．この問題の危険性を，ショッピングサイトを例として考える．サイト事業者が商品情報やサイトそのものをクラウドサービスを用いて構築することは近年珍しくない．この場合，利用者の名前や住所といった個人情報はもちろん，購入商品に関する情報もクラウドサービスのプロバイダも含め，サイト事業者以外には公開すべきではない．今，ある数名の利用者が問合せ $q_{i_1}, q_{i_2}, \cdots, q_{i_m}$ をサーバへ送信したとする．このショッピングサイトが EDB モデルを採用していた場合，保存されているタプルは暗号化されており，また問合せからタプルの中身を推定することもできない．しかし，この利用者集合が共通の問合せを送信したという事実は閲覧可能であり，Link disclosure の可能性は残っている．この利用者集合に属するある利用者が高額な商品ばかりを検索，購入していたことを公開すると，同一の集合に属する他の利用者も高額な商品を購入していたと推測可能である．その結果，攻撃者がこれらの利用者に対し重点的に攻撃を仕掛けてくる可能性がある．また，商品 X の購入を公開した複数の利用者に共通する問合せが q しか無かった場合，商品 X には問合せ q に用いられている索引値が対応していると分かってしまう．この事実は，索引の安全性が破られる要因となる．実際，既存の索引構築法では，利用者側の関係情報を用いた攻撃に関しては考慮されていないものが多い．

そこで，一つの問合せ条件に対して複数種類の問合せ表現 $q_1, q_2, \cdots$ を許すことを考える．利用者が問合せ毎に異なる問合せ表現を用いることで，同じ問合せを使用した利用者集合を求めることが困難になる．つまり，共通の問合せを用いたことによる Link disclosure を防ぐことが出来る．さらに，問合せごとに異なる問合せを用いる事で，どの問合せが重要であるのかも秘匿することが出来る．重要な問合せが攻撃者に漏洩することで，その問合せに対応するタプルを重点的に攻撃される危険性があるが，重要な問合せを秘匿することで，この危険性についても回避できる．我々は，この目的のために EDB モデル上で動作する新しい索引手法 (ベクトル索引) とそれを用いた問合せ手法を提案する．提案する索引は，汎用的な用途に利用できるようにバケット索引を基にしている．また，問合せに乱数を設定する次元を追加した高次ベクトルとして表現する．問合せごとに異なる値を取る乱数を追加することで，異なる問合せ表現を実現する．

2 暗号化データベースモデル

我々が対象とする EDB モデルでは，図 1 に示すようにデータ提供者 (data owner)，データ利用者 (data user)，サーバ (server)，攻撃者 (attacker) の 4 プレーヤが関係している．ただし，データ提供者と利用者が同じ場合もある．データ提供者のクライアントは，スキーマ R に

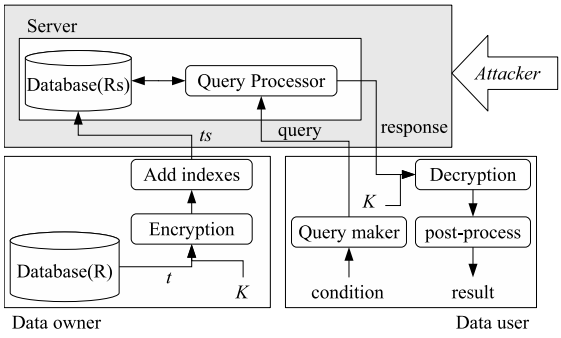


図 1: 暗号化データベースモデルにおける 4 プレーヤ．

おけるタプル t をサーバへ保存する時，暗号化と問合せ用索引 (index) 情報の付加を行う．そして，安全性の保証されたスキーマ R_s におけるタプル t_s をサーバへ送信する．データ利用者のクライアントは，利用者がサーバへ問い合わせる場合，問合せ条件から索引を用いた問合せを作成しサーバへ問い合わせる．また，サーバから返された暗号化タプルを復号し後処理を行い利用者が問い合わせたタプルのみを出力する．これらのクライアントには，予めスキーマ R_s に関する情報及び暗号鍵 K が知らされている．タプルがクライアント側であらかじめ暗号化されるので，サーバは暗号化されたデータのみを扱い，また問合せもデータを復号することなく行うことになる．4 番目のプレーヤである攻撃者は，サーバへ保存されている暗号化データやサーバへの通信を傍受するものと仮定する．攻撃者は，サーバへの通信を傍受することで，どの利用者がどんな問合せを行ったのかを知る事ができる．つまり利用者識別子を id, 問合せを q とすると，(id, q) の組からなる問合せログを取得することが可能である．ここでの利用者識別子とは，アプリケーションで使用されている利用者 ID や利用者の IP アドレスとする．これらを攻撃者から隠すことは難しい．したがって，各々の利用者がいったい何を問い合わせているのかを隠す必要がある．

3 ベクトル索引

我々が提案するベクトル索引では，問合せログからの Link disclosure を防ぐために，一つの問合せ条件に対して複数の表現を許す．問合せの度に異なる表現を用いることで，同じ条件で問い合わせた利用者集合を求めることが困難になる．ここでは，ベクトル索引の説明を始める前に，先ず EDB モデルで使用されるバケット索引について説明する．ある属性 A に対するバケット索引を計算するには，対象属性 A の値域 D_A を N 個の区間に分割する必要がある．この分割には既存の手法を用いることができる．分割された区間は，整数ラベル $b_1, b_2, \cdots, b_N$

が順に設定される．このとき，対象属性 A に全順序が設定されており範囲問合せを行う場合には，順保存のラベル付けを使用する必要がある．つまり，それぞれ b_i, b_j とラベル付けされた二つの区間 S_i, S_j が $S_i < S_j$ ならば $b_i < b_j$ となる．サーバ上のテーブル R_s へ保存される暗号化タプル t_s は，その属性 A の値が含まれる区間のラベルを索引 I の値として持つことになる．対象属性の値域 D_A がどのように分割されラベル付けされているかは，この暗号化テーブル R_s を使用するすべてのクライアントに予め通知される．属性 A の値が c であるタプルを問い合わせる場合，クライアントは先ず c がどの区間に含まれるのかを調べる．サーバへ送信される問合せは，その区間のラベルを b_q とすると $\{t_s | t_s \in R_s \wedge t_s.I = b_q\}$ となる．このバケット索引を用いた問合せ処理では，属性 A の値がラベル b_q で表される区間に含まれるすべてのタプルがサーバから問合せ結果として返される．そのため，クライアントはそれら問合せ結果を復号し，本当に属性 A の値が c であるものだけを取り出す後処理が必要となる．バケット化により，複数の属性値に対して共通のラベルを索引値とすることで，どの属性値がどれだけ使用されているのかという属性値の頻度情報を隠すことができる．我々のベクトル索引でも，属性値の使用頻度を隠すために，バケット化を使用する．

さて，このバケット索引を用いた問合せ $\{t_s | t_s \in R_s \wedge t_s.I = b_q\}$ における条件項 $t_s.I = b_q$ は，ベクトルの内積として，

$$< (1 \quad -t_s.I)^t, (b_q \quad 1)^t > = (1 \quad -t_s.I)(b_q \quad 1)^t = 0$$

と書くことも出来る．この様に記述すると，一つ目のベクトル $\mathbf{i} = (1 \quad -t_s.I)^t$ はタプル t_s によって定まり，二つ目のベクトル $\mathbf{q} = (b_q \quad 1)^t$ は問合せによって定まる．従って，サーバが行う問合せ処理とは，与えられた問合せベクトル $\mathbf{q}$ に直交する索引ベクトル $\mathbf{i}$ を持つタプル t_s を探すことと考えることが出来る．ある問合せ条件に対して複数の表現，つまり複数の問合せベクトルが作成できるためには，ベクトルの次元を増やせば良い．実際，我々のアプローチで使用する問合せベクトルは $\mathbf{q} = (b_q \quad p \quad 1)^t$ である．ここで，p は乱数であり，問合せごとに異なるランダムな値である．この問合せベクトルに合わせて，タプル t_s へ付加する索引ベクトルは $\mathbf{i} = (1 \quad 0 \quad -t_s.I)^t$ となる．問合せベクトルが一致するためには，問い合わせるバケットのラベル b_q と乱数 p の両方が一致する必要がある．つまり，一致の確率は同じ乱数が使用される確率よりも低い．もちろん，この問合せベクトルから各利用者が何を問い合わせたのか，問合せ条件 b_q を得ることは容易である．そのため，問合せベクトル $\mathbf{q}$ の各成分が何を表しているのか秘匿する必要がある．ベクトルにおいて各成分の意味を変えるためには，[6] でも用いられているように，一般的に行列を乗じて基底を変えることが考えられる．そこで，3 次正則正方行列 M を乗じた $\hat{\mathbf{q}} = M(b_q \quad p \quad 1)^t$ を問合せベクトルとして用いることにする．合わせて，この問合せベクトルとの内積計算を行うために索引ベクトルには，M の逆行列を乗じた $\hat{\mathbf{i}} = ((1 \quad 0 \quad -t.I)M^{-1})^t$ を使用する．このとき，

$$< \hat{\mathbf{i}}, \hat{\mathbf{q}} >= \hat{\mathbf{i}}^t \hat{\mathbf{q}} = (1 \quad 0 \quad -t.I)M^{-1}M(b_q \quad p \quad 1)^t$$
$$(1 \quad 0 \quad -t.I)(b_q \quad p \quad 1)^t = b_q - t.I$$

となる．したがって，上述の $t_s.I = b_q$ という条件項は，このベクトル索引を用いると $< \hat{\mathbf{i}}, \hat{\mathbf{q}} >= 0$ と書くことができる．よって，問合せは，$\{t_s | t_s \in R_s \wedge < \hat{\mathbf{i}}, \hat{\mathbf{q}} >= 0\}$ となる．

そのため問合せベクトル $\mathbf{q}$ を暗号化する．この暗号化は，問合せ条件 b_q を秘匿することはもちろん，暗号化されたままサーバ上での問合せ処理，すなわち内積計算が行える必要がある．今回我々は，この目的に合う暗号化方法として，正則行列を用いたベクトルのための非対称暗号 [6] を使用した．この暗号方法は，同じ次元数の二つのベクトル集合 U, V に含まれるベクトル $\mathbf{u} \in U, \mathbf{v} \in V$ をそれぞれ別の方法で暗号化し，暗号化したまま内積 $< \mathbf{u}, \mathbf{v} >$ が計算できる暗号化方法である．また，同じ集合に含まれるベクトルどうしの内積 $< \mathbf{u}_1, \mathbf{u}_2 > (\mathbf{u}_1, \mathbf{u}_2 \in U)$ 及び $< \mathbf{v}_1, \mathbf{v}_2 > (\mathbf{v}_1, \mathbf{v}_2 \in V)$ が計算出来ないという特徴もある．暗号化はベクトルと同じ次元の正則正方行列 M を暗号鍵として次のように行う．ベクトル $\mathbf{u} \in U$ の暗号化ベクトル $\mathbf{u}'$ は，$\mathbf{u}' = M^t\mathbf{u}$，$\mathbf{v} \in V$ の暗号化ベクトル $\mathbf{v}'$ は，$\mathbf{v}' = M^{-1}\mathbf{v}$ となる．このとき，異なる集合に含まれる暗号化ベクトルどうしの内積は，

$$< \mathbf{u}', \mathbf{v}' >= \mathbf{u}'^t\mathbf{v}' = (M^t\mathbf{u})^t M^{-1}\mathbf{v}$$
$$= \mathbf{u}^t M M^{-1}\mathbf{v} = \mathbf{u}^t\mathbf{v} =< \mathbf{u}, \mathbf{v} >$$

となり計算出来る．この暗号を用いると，M を暗号鍵である 3 次正則正方行列として，索引ベクトルは $M^t(1 \quad 0 \quad -t.I)^t$，問合せベクトルは $M^{-1}(b_q \quad p \quad 1)^t$ となる．同じ集合に属する暗号化ベクトルどうしの内積が計算できないという特徴から，サーバ上の索引ベクトルから暗号鍵である行列 M の再計算を防げる．

3.1 ベクトル索引を用いた問合せ処理

EDB モデルにおいて，属性 A に対する索引としてベクトル索引を用いるためには，バケット索引と同様に，対象属性の値域 D_A を N 区間に分割する必要がある．そして，分割された各区間には，整数ラベル $b_1, b_2, \cdots, b_N$ が順に設定される．このとき，属性 A の値がラベル b の区間に含まれるタプル t_s におけるベクトル索引の値は，$\mathbf{i} = ((1 \quad 0 \quad -b)M^{-1})^t$ となる．

このベクトル索引を用いて属性 A の値が c であるタプルを求める場合，クライアントは先ず c が含まれる区間のラベル b_q を計算する．そして，問合せベクトル $\mathbf{q} = M(b_q \quad p \quad 1)^t$ を用いて，問合せ $\{t_s|t_s \in R_s \wedge < t_s.\mathbf{i}, \mathbf{q} > = 0\}$ をサーバへ送信する．なお，問合せベクトル $\mathbf{q}$ における p は乱数であり，問合せごとに異なるランダムな値を設定する．この乱数に同じ値が使用される確率は小さいものとする．サーバから返された結果を復号し属性 A の値が実際に c であるタプルを取り出す後処理は，従来のバケット索引を用いた問合せと同じである．

属性 A の値が，c_m から $c_M(c_m < c_M)$ の間に含まれるタプルを求める場合，この場合もクライアントは c_m, c_M が含まれる区間のラベルを計算する．c_m と c_M がラベルが b である共通の区間に含まれているならば，一致問合せの場合と同様に問合せベクトルを $\mathbf{q} = M^{-1}(b_q \quad p \quad 1)^t$ として，$\{t_s|t_s \in R_s \wedge < t_s.\mathbf{i}, \mathbf{q} > = 0\}$ を問合せとする．c_m, c_M がそれぞれラベル $b_m, b_M(b_m < b_M)$ である区間に含まれているのであれば，問合せベクトルとして，

$$\mathbf{q} = M(\frac{b_m + b_M}{b_m - b_M} \quad p \quad \frac{2}{b_m - b_M})^t$$

を用いる．このとき，索引ベクトルと問合せベクトルの内積は，

$$< t_s.\mathbf{i}, \mathbf{q} >= (1\ 0\ -b)M^{-1}M(\frac{b_m + b_M}{b_m - b_M}\ p\ \frac{2}{b_m - b_M})^t$$
$$= (1\ 0\ -b)(\frac{b_m + b_M}{b_m - b_M}\ p\ \frac{2}{b_m - b_M})^t = \frac{b_m + b_M - 2b}{b_m - b_M}$$

である．したがって，問合せとして $\{t_s|t_s \in R_s \wedge -1 \leq\ < t_s.\mathbf{i}, \mathbf{q} >\ \leq 1\}$ を用いることで，

$$\begin{aligned} -1 \leq\ & < t_s.\mathbf{i}, \mathbf{q} > & \leq 1 \\ -1 \leq\ & \frac{b_m + b_M - 2b}{b_m - b_M} & \leq 1 \\ -(b_m - b_M) \leq\ & b_m + b_M - 2b & \leq b_m - b_M \\ b_m \leq\ & b & \leq b_M \end{aligned}$$

より，目的の範囲に含まれる暗号化タプルを取得することができる．どちらの場合も，クライアントはサーバから返された結果を復号し属性 A の値が実際に c_m から c_M の間に含まれるタプルを取り出す後処理を行う．

クライアントは，このベクトルを用いた問合せ，$\{t_s|t_s \in R_s \wedge -1 \leq\ < t_s.\mathbf{i}, \mathbf{q} >\ \leq 1\}$ をサーバへ送信する．この問合せにおける内積部分は，

$$\begin{aligned} < t_s.\mathbf{i}, \mathbf{q} > &= (1\ 0\ -b)MM^{-1}(\frac{b_m + b_M}{b_m - b_M}\ p\ \frac{2}{b_m - b_M})^t \\ &= (1\ 0\ -b)(\frac{b_m + b_M}{b_m - b_M}\ p\ \frac{2}{b_m - b_M})^t \\ &= \frac{b_m + b_M}{b_m - b_M} - \frac{2b}{b_m - b_M} \end{aligned}$$

と計算できる．したがって，条件部分は，

$$\begin{aligned} -1 \leq\ & < t_s.\mathbf{i}, \mathbf{q} > & \leq 1 \\ -1 \leq\ & \frac{b_m + b_M}{b_m - b_M} - \frac{2b}{b_m - b_M} & \leq 1 \\ -(b_m - b_M) \leq\ & b_m + b_M - 2b & \leq b_m - b_M \\ b_m \leq\ & b & \leq b_M \end{aligned}$$

となり，ラベル b_m から b_M の区間に含まれているタプルだけを取得していることが分かる．なお，サーバより返された結果に対しクライアントが復号化とフィルタリングを行うことはこれまでと同じである．

3.2 安全性の考察

ベクトル索引を用いた問合せが一致し，ユーザの関係が漏洩する可能性について考える．乱数が取り得る値を N_p 種類とすると，同じバケットラベル b を問い合わせる二つの問合せが一致する可能性は，乱数が一致する確率に等しく，$1/N_p$ である．問合せが θ 回一致した場合に，ユーザ間の関係が推定されるとすると，その確率は $(1/N_p)^\theta$ となる．今，乱数に 32bit の整数値を用いたとすると，$N_p \approx 4.3 \times 10^9$ であり，$\theta = 2$ としても，約 6.2×10^{-20} である．

4 評価実験

本稿で提案するベクトル索引によって Link disclosure の可能性がどの程度抑制できるのか，及びサーバ上での計算時間はどの程度増加するのかを定量的に評価するためにシミュレーション実験を行った．実験に用いたデータセットは共著関係データ[1]中の Bibliography on database systems である．このデータセットには 20,011 件の論文と 8,397 名の著者が含まれている．本評価実験では，一つの論文情報を一つのタプルとしてサーバ上の暗号化データベースに保存し著者を利用者とした．すなわち，実験に用いたデータベースには 20,011 個の暗号化タプルが保存されており，8,397 名の利用者が問い合わせを行う事になる．その上で，利用者である著者が自身の執筆した論文をタイトルを用いて一致検索を行うというシナリオで問合せをシミュレーションした．また，論文タイトルに辞書順を設定し，利用者である著者が自身の論文を含む範囲を検索したというシナリオで範囲問合せのシミュレーションも行った．使用した索引手法は，本稿で我々が提案したベクトル索引と範囲問合せが可能なバケット索引の二種類である．両索引の作成に必要な対象ドメインを N 個のバケットに分割する方法としては，以下の二種類の手法を実装した．

平均分割法 (avg) 対象となるドメインを均等の幅を持つ N 個のバケットに分割する方法．どのようなタ

[1] http://liinwww.ira.uka.de/bibliography/

プルが保存されるのか分かってない場合，ナイーブにはこのように均等に分割する方法が使用される．

最適分割法 (opt) すべてのタプルの値を基に，各バケットに格納されるタプル数が均等になるように分割する方法．予めタプル関する情報が必要となるため静的なデータベースにのみ適用可能である．

バケットの数 N は，500, 1,000 及び 2,000 の三種類を用いた．サーバへ保存されるタプル数である論文数が約 20,000 であるため，作成されたバケットにはそれぞれの場合についておよそ 40, 20 及び 10 個のタプルが格納されることになる．なお，ベクトル索引におけるランダムダミーデータには 32bit 整数を使用した．そのため，ランダムダミーデータは約 40 億通り整数から一様な分布で値が選ばれることになる．

先ず，提案手法と既存手法を用いた場合に Link disclosure の危険性がどれほど変化するかについて，次の関係発見手法を用いて評価した．これは，サーバに残された問合せログに対して，共通の問合せを閾値 θ 回以上行った利用者に関係があると推定するものである．この手法により推定された関係と実際の共著関係を比較し，精度と再現率を計算した．図 2 は横軸に閾値を縦軸に精度及び再現率を取ったグラフである．閾値の括弧内はバケットの数である．凡例の avg 及び opt はバケットの作成がそれぞれ平均分割法，最適分割法を使用していることを表しており，vec はベクトル索引を使用していることを表す．既存のバケット索引を用いた場合，閾値 θ により精度と再現率の間にトレードオフ関係があることが見て取れる．逆に我々が提案するベクトル索引を用いた場合は精度，再現率ともに閾値によらず 0 である．これは，今回の実装においてランダム値の取り得る範囲が約 40 億通りあり問合せ数に比べ十分大きいため，すべての問合せが異なる値としてサーバへ送られたためである．したがって，どの利用者が同じ条件で問い合わせたのかを隠すだけでなく，どの条件での問合せ頻度が多いかについても完全に秘匿できているといえる．

次に，問合せに要する平均時間を計測した．図 3 は，横軸に使用した手法と縦軸に要した時間をミリ秒単位で取ったものである．手法の opt 及び avg は先程と同様使用したバケット分割手法を表しており，括弧内の数値はバケットの数すなわち対象となるドメインをいくつの領域に分割したのかを表している．また，凡例における pre, server 及び post の意味は次の通りである．

事前準備時間 (pre) サーバへ問い合わせる前にクライアント上で必要となる処理にかかる時間．具体的には，問合せ条件から使用する索引手法に対応した問合せを作成するのに要する時間である．

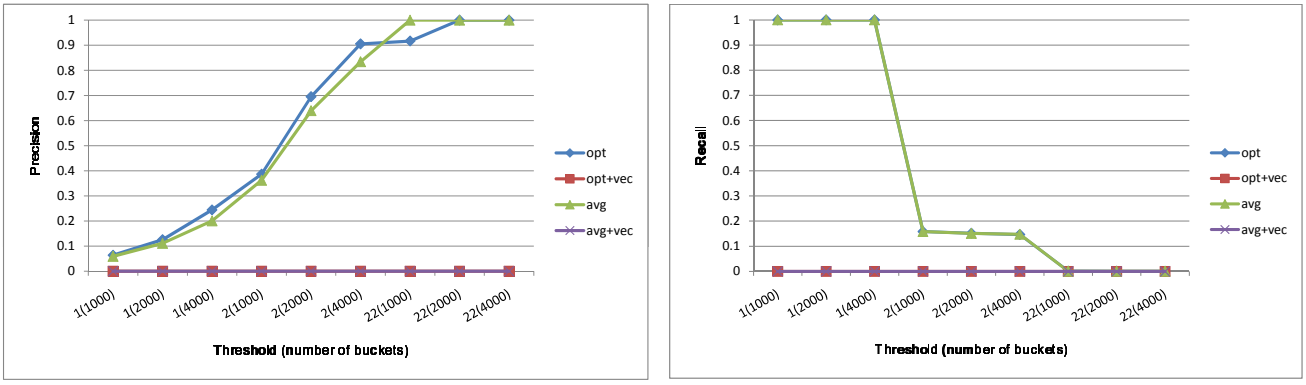
サーバ処理時間 (server) サーバ上で問合せ処理に要する時間．クライアントとの通信時間も含む．クライアントから送られた問合せをサーバが評価する時間と，クライアントがその結果を受け取るのに要する時間の合計である．

後処理時間 (post) サーバからの問合せ結果に対してクライアントが行う後処理にかかる時間．具体的には，問合せ結果を復号し問合せ条件に適合するタプルのみを取り出す処理に要する時間である．

図では，事前準備に要した時間 pre がほとんど見えなくなっており，どの場合においても，問合せ条件から EDB モデルのための問合せを作成する時間は十分小さく無視できると言える．次に，サーバ上での処理時間は，どのバケット数でもベクトル索引の方が遅い．これは，既存のバケット索引が値の単純な比較だけで済むのに対し，提案手法ではベクトルの内積を計算しなければならず時間がかかっていると言える．また，バケット数の増加に応じてサーバ上での処理時間が短くなっている．これは，各バケットに含まれるタプル数が変化するためで，バケット数が多ければ，各バケットに含まれるタプル数は小さくなる．つまり，問合せの結果として返却されるデータ量が少なくて済む．そのため，要する時間が少なく済んでいる．最後に，後処理に要した時間 $post$ であるが，これはバケット索引とベクトル索引の間で差は見られなかった．ただし，バケット数の増加に伴い小さくなった．これも先ほどと同様，バケット数が大きくなることで返却されるタプル数は減少する．したがって，後処理として復号及びフィルタリングを行わなければならないタプル数が少なくて済んでいるといえる．

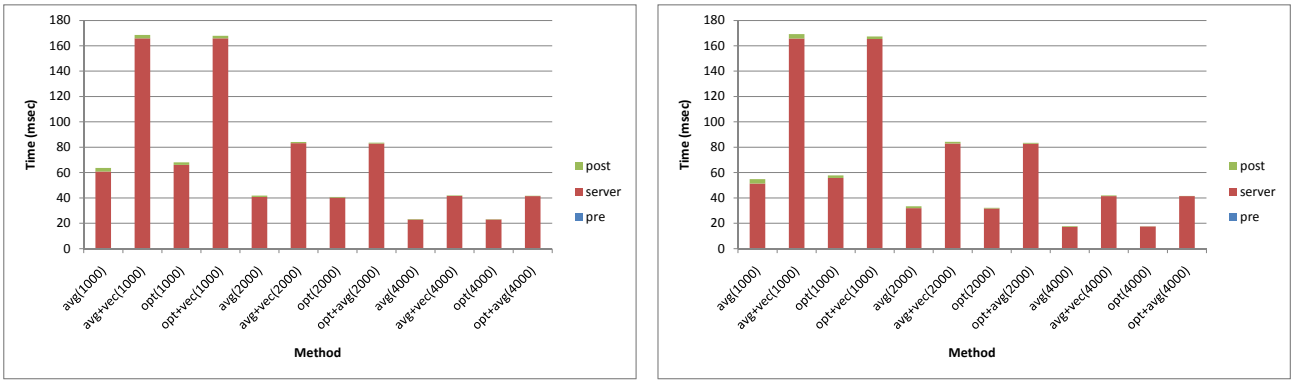
5 おわりに

本稿では，暗号化データベースモデルにおいて，問合せからユーザ間の関係情報を推測する Link disclosure の発生を防ぐ，新しい索引手法であるベクトル索引とそれを用いた問合せ処理方法を提案した．ベクトル索引では，$I = c$ という条件式はベクトルの内積を用いて $(1\ -I)(c\ 1)^t = 0$ と表される．条件式の表現方法としてベクトルを用いる事で，次元の追加と基底の変換が可能となる．その結果，一つの問合せ条件を複数の条件式として表現することが可能となり，どの利用者が同じ条件で問い合わせているのかを秘匿することが可能となる．すなわち，問合せからの Link disclosure を防ぐことが可能である．評価実験では，既存索引手法に比べてサーバ上での問合せ実行速度は遅くなるが，共通の問合せ条件を用いている利用者を求める事が困難であることが示された．今後の課題としては，問合せだけでなく問合せ



(a) 精度　(b) 再現率

図 2: 閾値を変化させた場合における関係発見手法の精度及び再現率.



(a) 一致問合せ　(b) 範囲問合せ

図 3: 問合せ処理に要した時間.

結果からの Link disclosure を防ぐためにこのベクトル索引の改良を行う予定である.

参考文献

[1] Evelyn Balfe and Barry Smyth. An analysis of query similarity in collaborative web search. In *BCIR 2005: Proceedings of the 27th European Conference on IR Research*, pp. 330–344, March 2005.

[2] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 216–227, New York, NY, USA, 2002. ACM.

[3] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pp. 720–731. VLDB Endowment, 2004.

[4] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 93–106, New York, NY, USA, 2008. ACM.

[5] Zheng-Fei Wang, Jing Dai, Wei Wang, and Bai-Le Shi. Fast query over encrypted character data in database. *Computational and Information Science*, pp. 1027–1033, 2005.

[6] Wai K. Wong, David Wai lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pp. 139–152, New York, NY, USA, 2009. ACM.